

FIG. 1

2/17

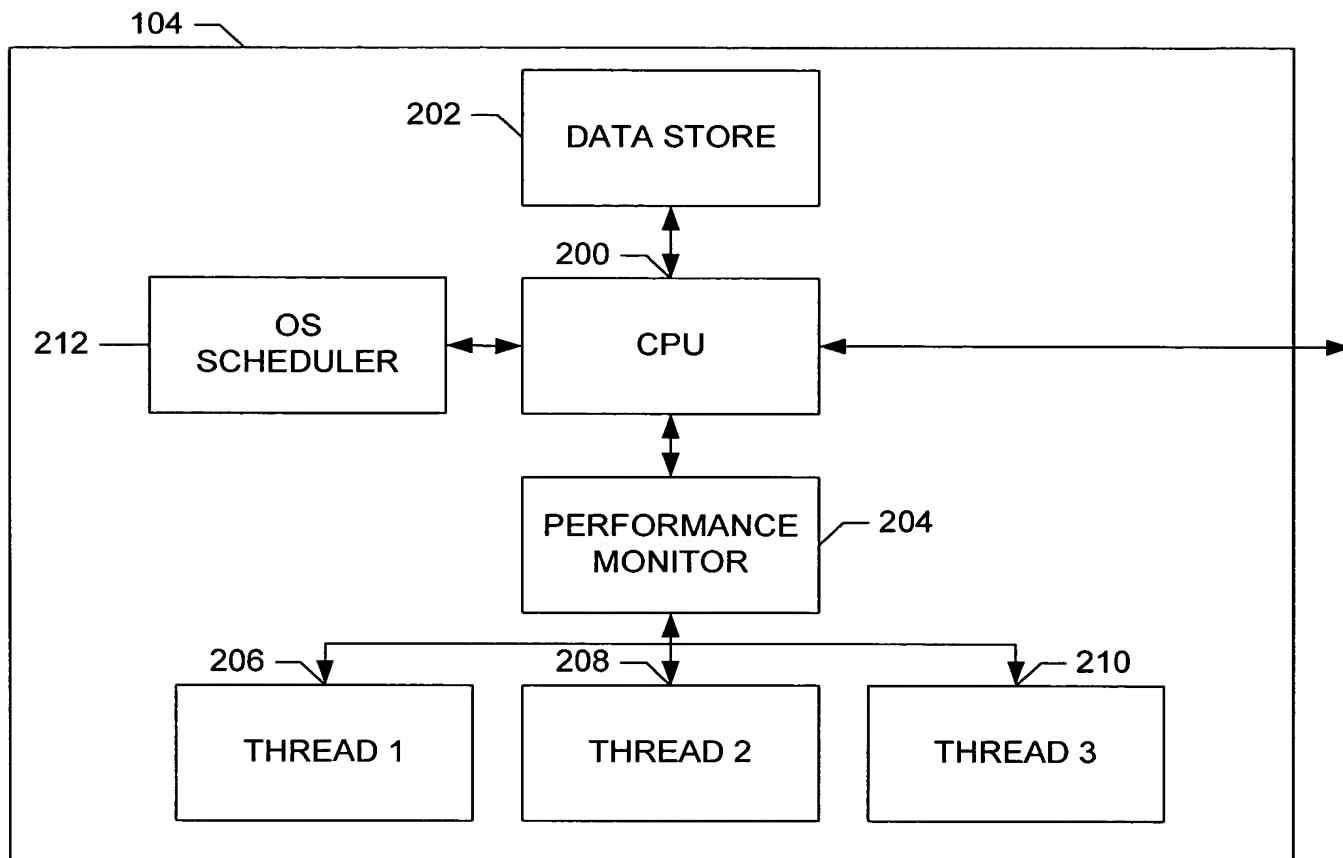


FIG. 2

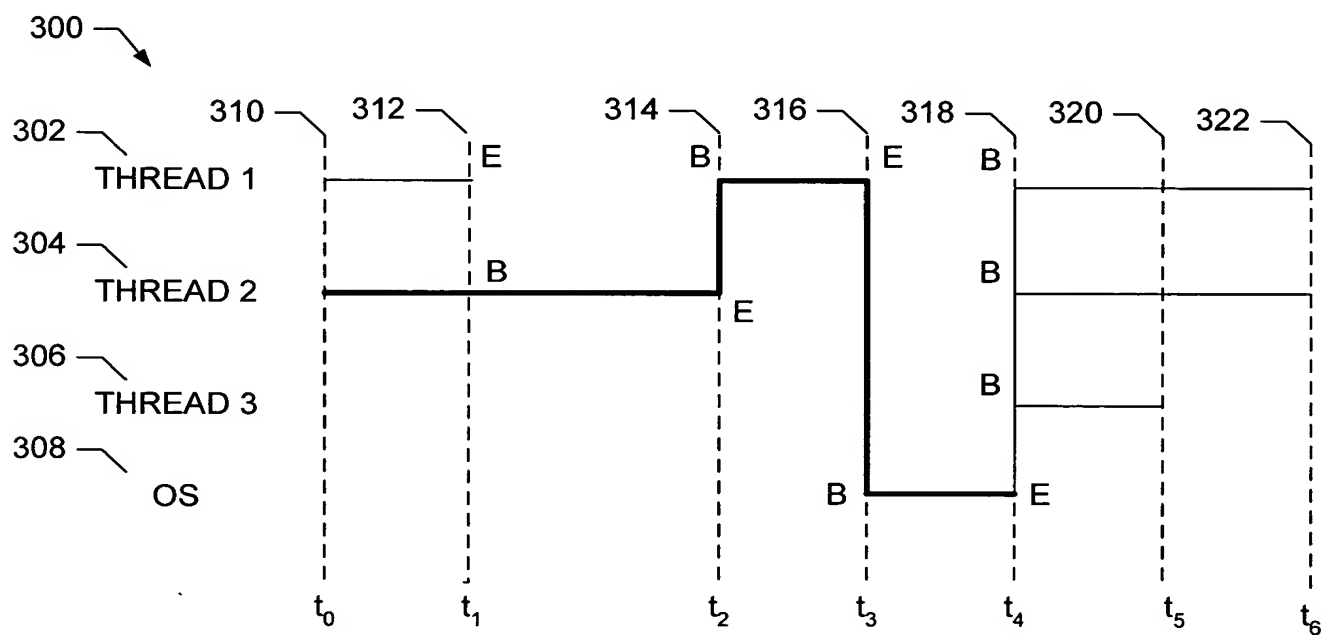


FIG. 3

3/17

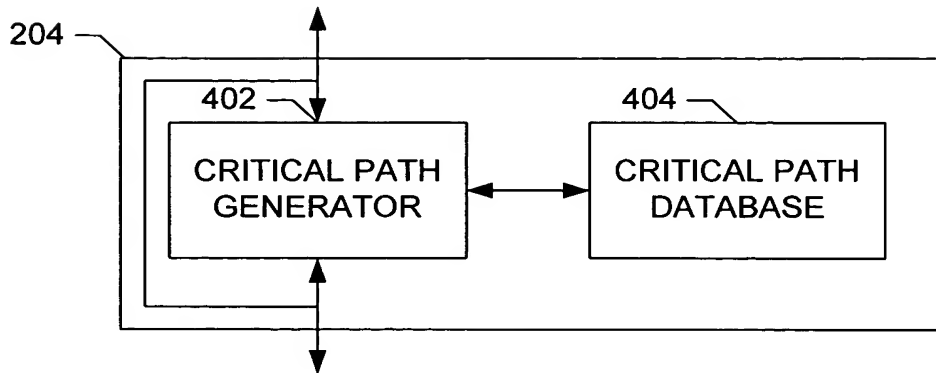


FIG. 4

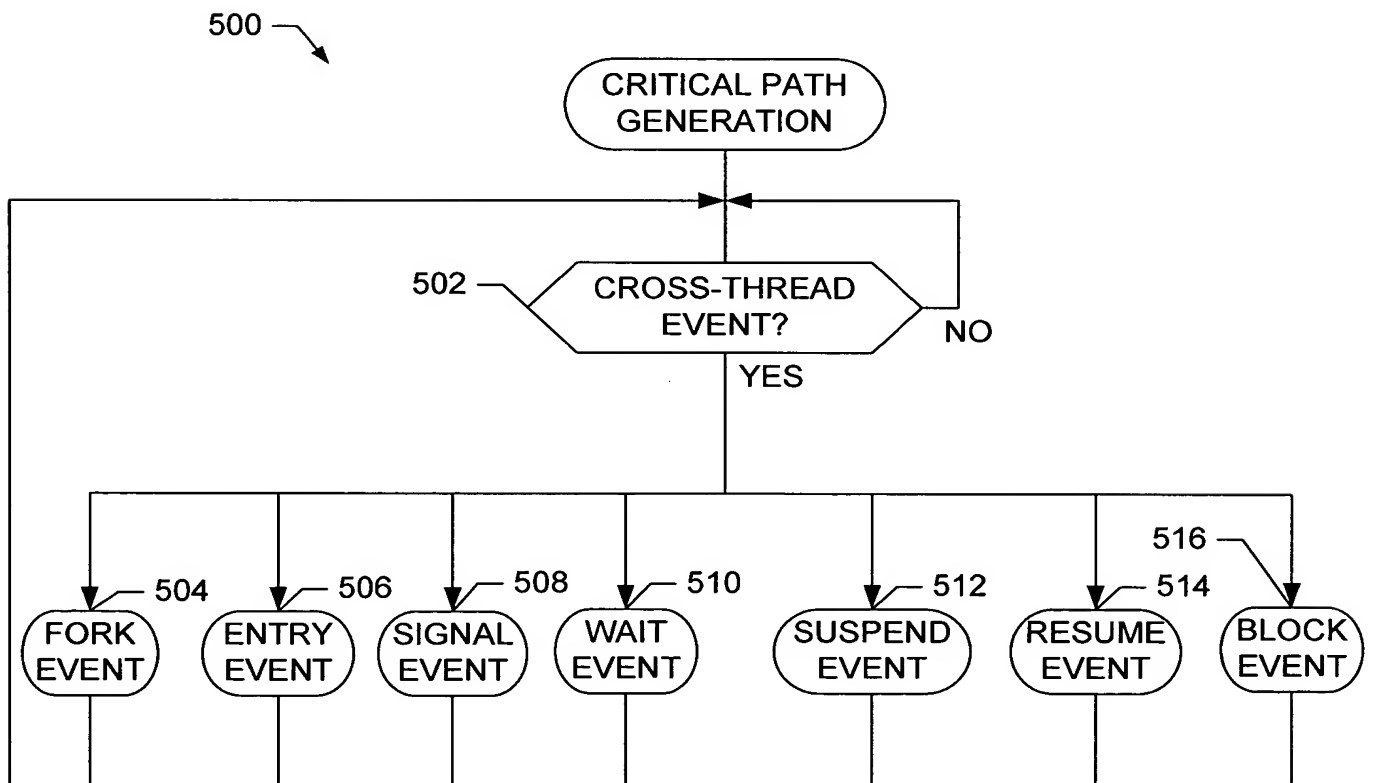



FIG. 5

4/17

600 

**FORK**

- \* CREATE NEW CHILD THREAD OBJECT
- \* CREATE NEW LEAVES FOR PARENT THREAD AND CHILD THREAD
  - ATTACH CHILD THREAD'S LEAF AS A PENDING LEAF TO THE CHILD THREAD
  - ATTACH PARENT THREAD'S NEW LEAF AS NEW LEAF FOR PARENT THREAD
- \* EXECUTE CREATE API
- \* IF CREATE FAILED
  - REMOVE CHILD LEAF & DELETE IT

FIG. 6

5/17

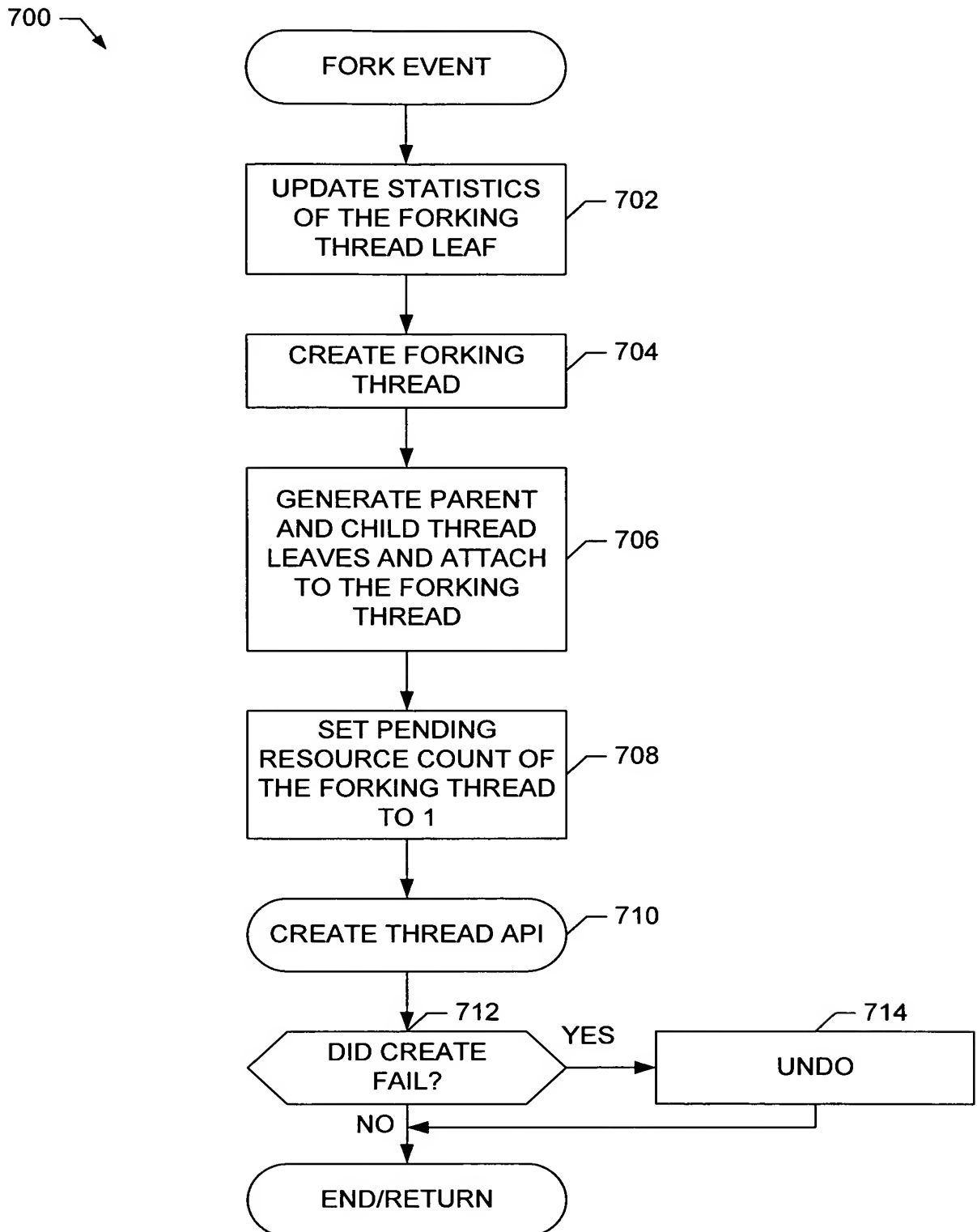


FIG. 7

6/17

800 ↘

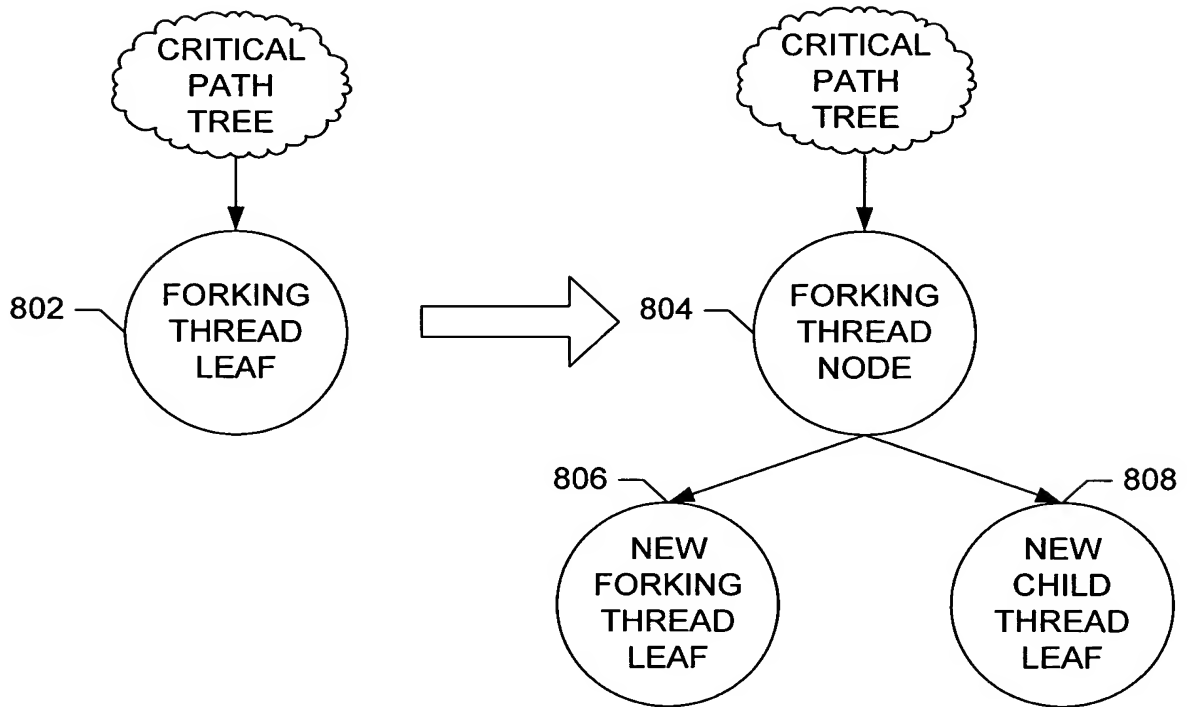


FIG. 8

900 ↘

**ENTRY**

- \* INCREMENT CONCURRENCY LEVEL
- \* DID WE MISS THE FORK CALL?
  - CREATE LEAF FOR CHILD BUT DO NOT ATTACH TO ANY PARENT NODE
  - RETURN
- \* UPDATE STATISTICS OF CHILD THREAD'S LEAF

FIG. 9

7/17

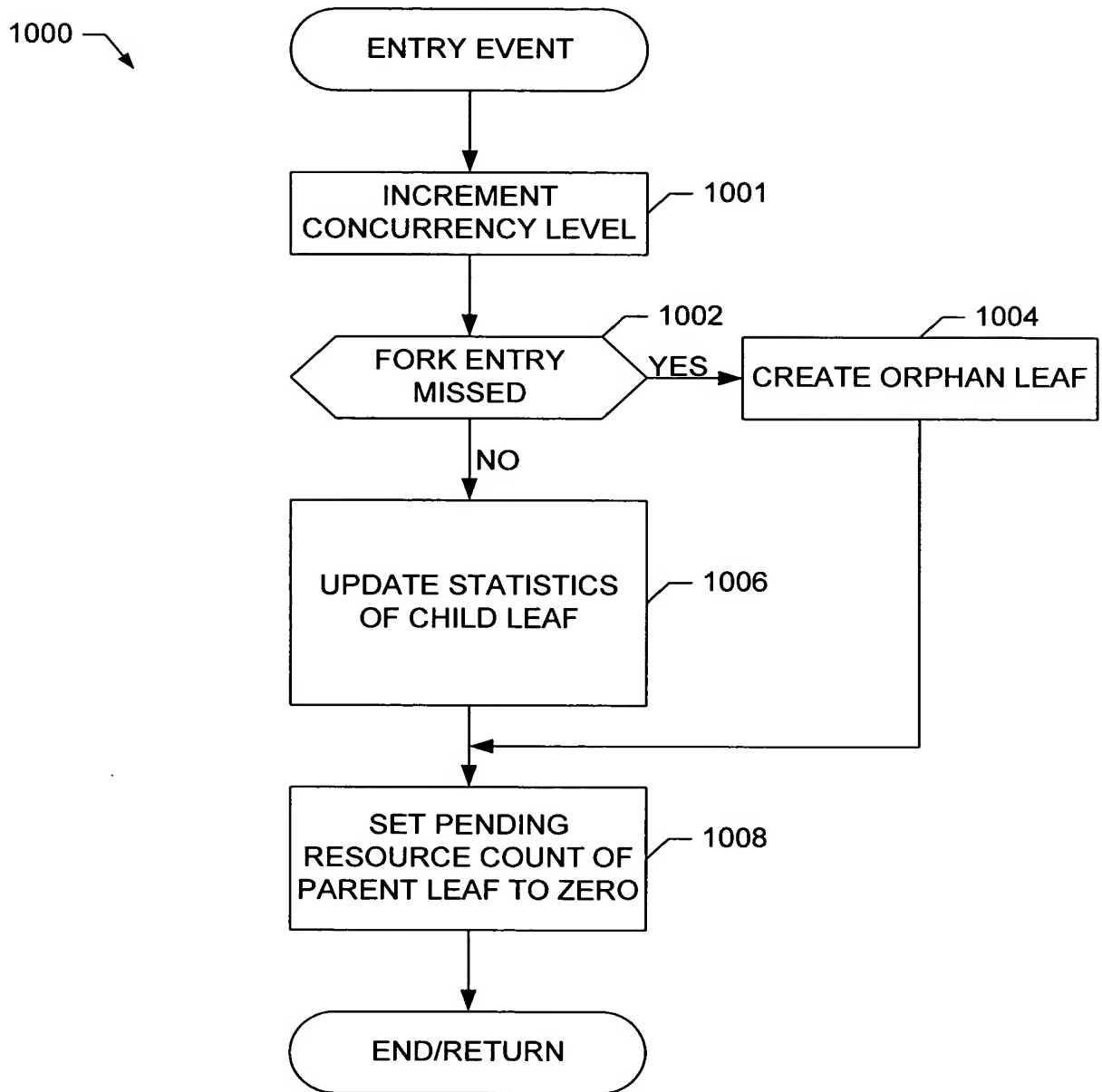


FIG. 10

8/17

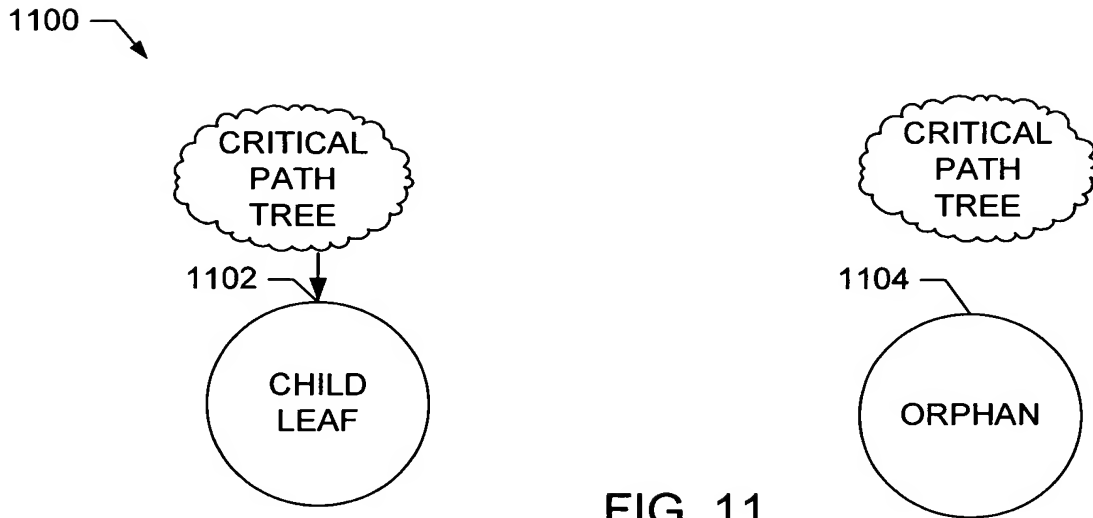


FIG. 11

1200

# **SIGNAL**

- \* GET THE NUMBER OF WAITING THREADS ON SYNC OBJECT
- \* IF NOT SELF TERMINATING DO THE API CALL
- \* IF SIGNAL WAS SUCCESSFUL
  - IF THERE WAS AT LEAST ONE WAITING THREAD FOR THIS OBJECT
    - GET THE CURRENT LEAF L OF THE SIGNALING THREAD
    - CREATE NEW LEAF S1 FOR SIGNALING THREAD W/ L AS PARENT NODE
    - CREATE PENDING NODE S2 FOR SIGNED THREAD W/ L AS PARENT NODE
    - SET RESOURCE CNT OF S2 TO THE COUNT OF RESOURCES BEING SIGNED
    - SET TIMESTAMP OF S2 TO THE CURRENT TIME
    - IF SIGNED OBJECT IS A SEMAPHORE
      - IF OBJECT ALREADY HAS A PENDING NODE WITH INFINITE SIGNAL CNT
        - REMOVE S2
      - ELSE
        - APPEND S2 TO PENDING NODE LIST OF SEMAPHORE
    - ELSE /\* NOT A SEMAPHORE \*/
      - IF THE SIGNED OBJECT ALREADY HAS ANOTHER PENDING NODE
        - REMOVE S2
      - ELSE
        - ADD S2 TO THE PENDING NODE LIST OF THE SIGNED OBJECT
    - ELSE /\* NO WAITING THREAD \*/
      - IF SIGNED OBJECT IS A SEMAPHORE
        - ADD SIGNAL CNT TO THE TOTAL PENDING RESOURCE COUNT OF SEMAPHORE SYNC OBJECT
  - IF THIS IS A THREAD TERMINATION OPERATION
    - IF THE TARGET THREAD WAS ACTIVE
      - DECREMENT CONCURRENCY LEVEL
    - SET THREAD STATE TO DEAD
    - IF THERE WAS NO WAITING THREAD
      - DELETE LEAF NODE OF TERMINATED THREAD
- \* IF SELF TERMINATING
  - CALL ACTUAL API NOW
- \* IF SIGNAL&WAIT, CALL WAIT () ENTRY

FIG. 12

9/17

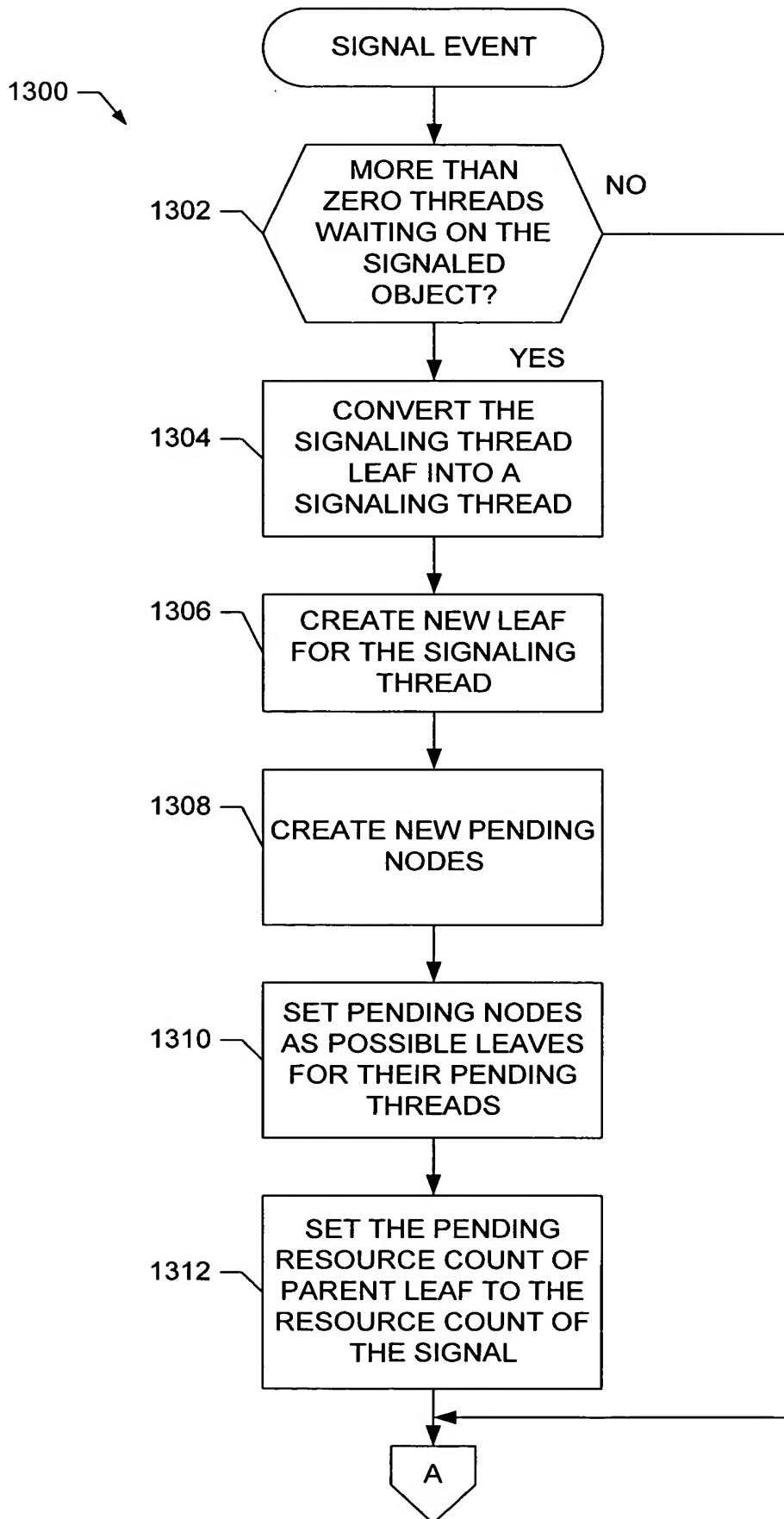


FIG. 13A

10/17

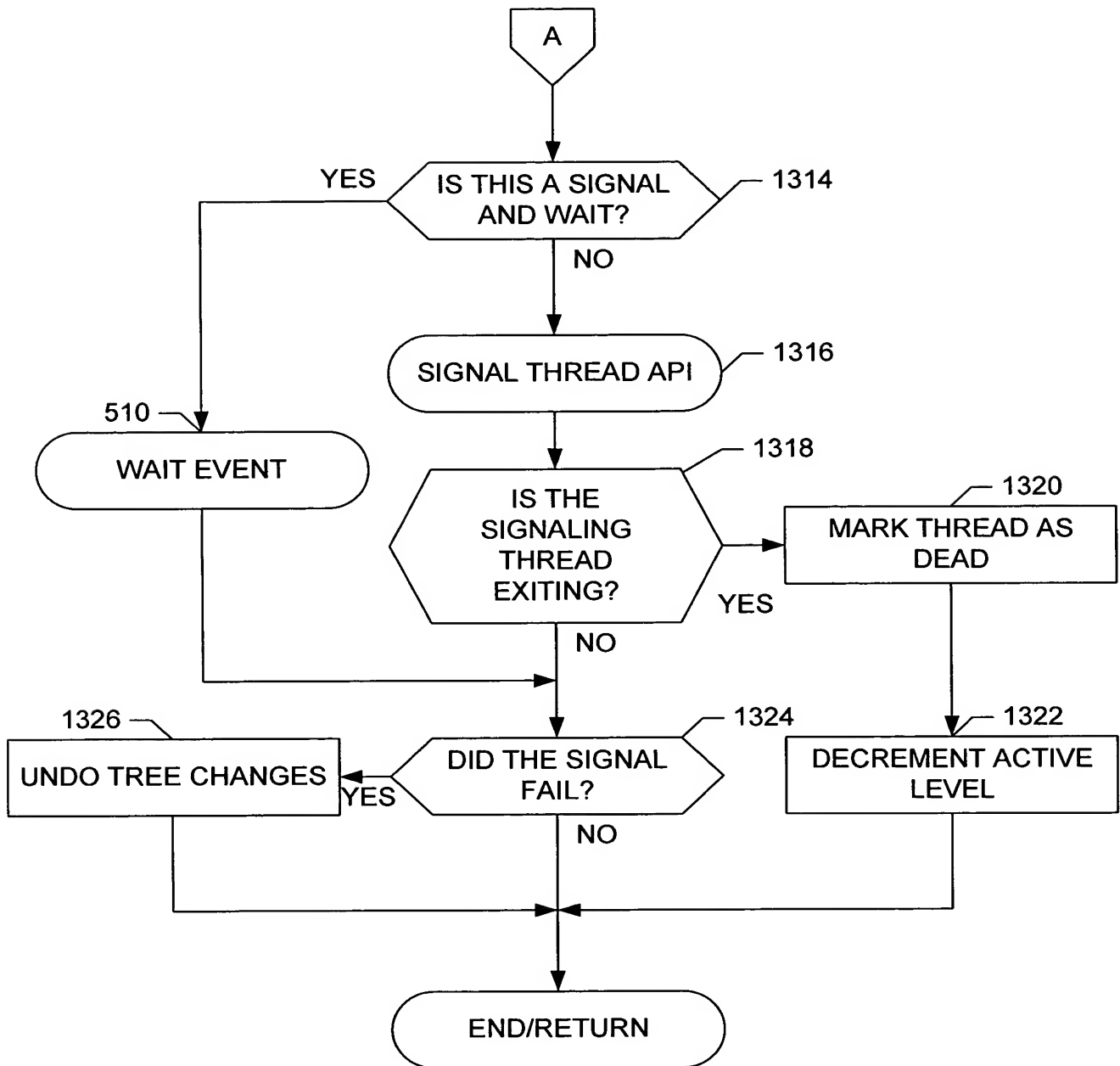


FIG. 13B

11/17

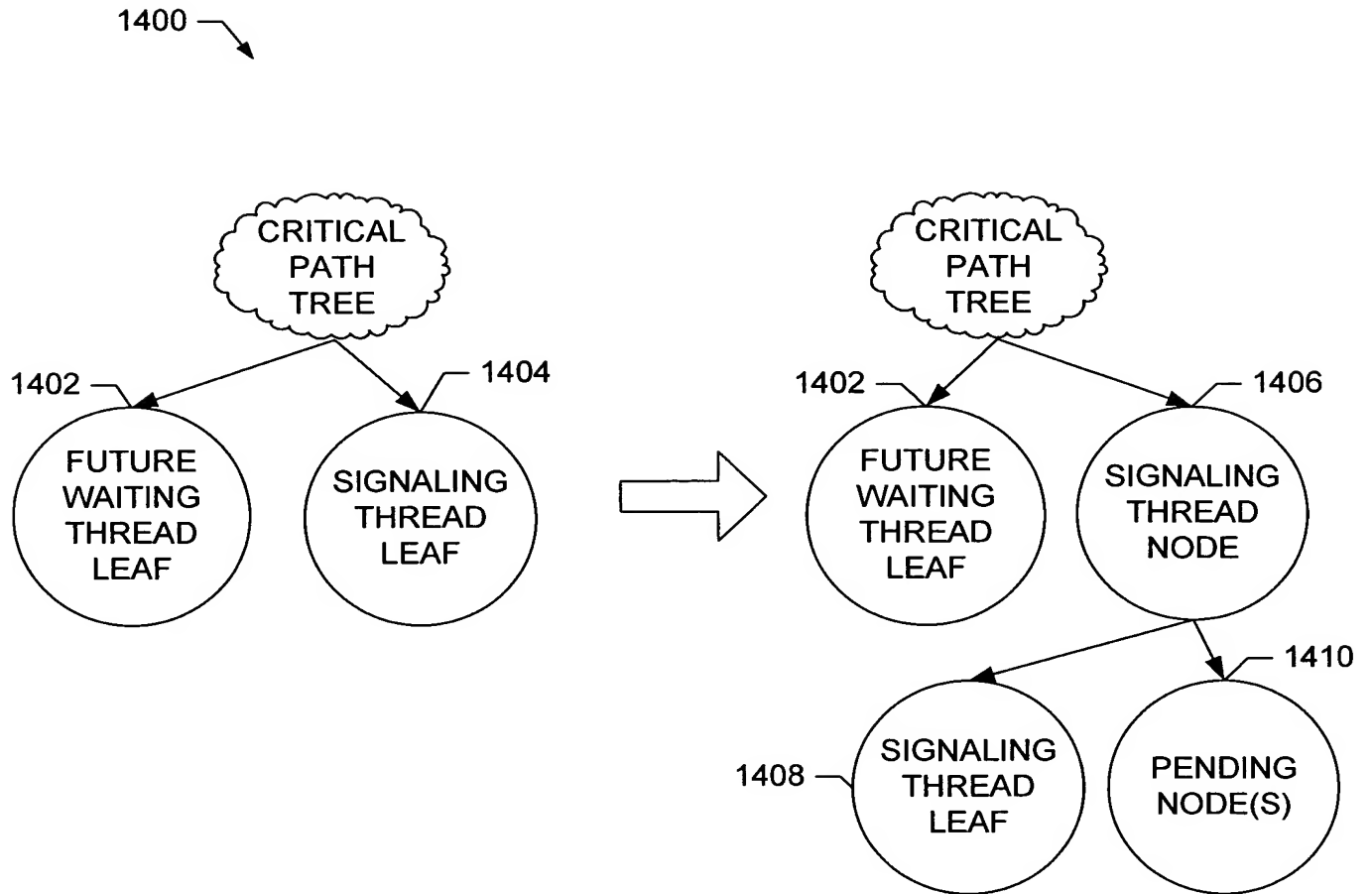



FIG. 14

12/17

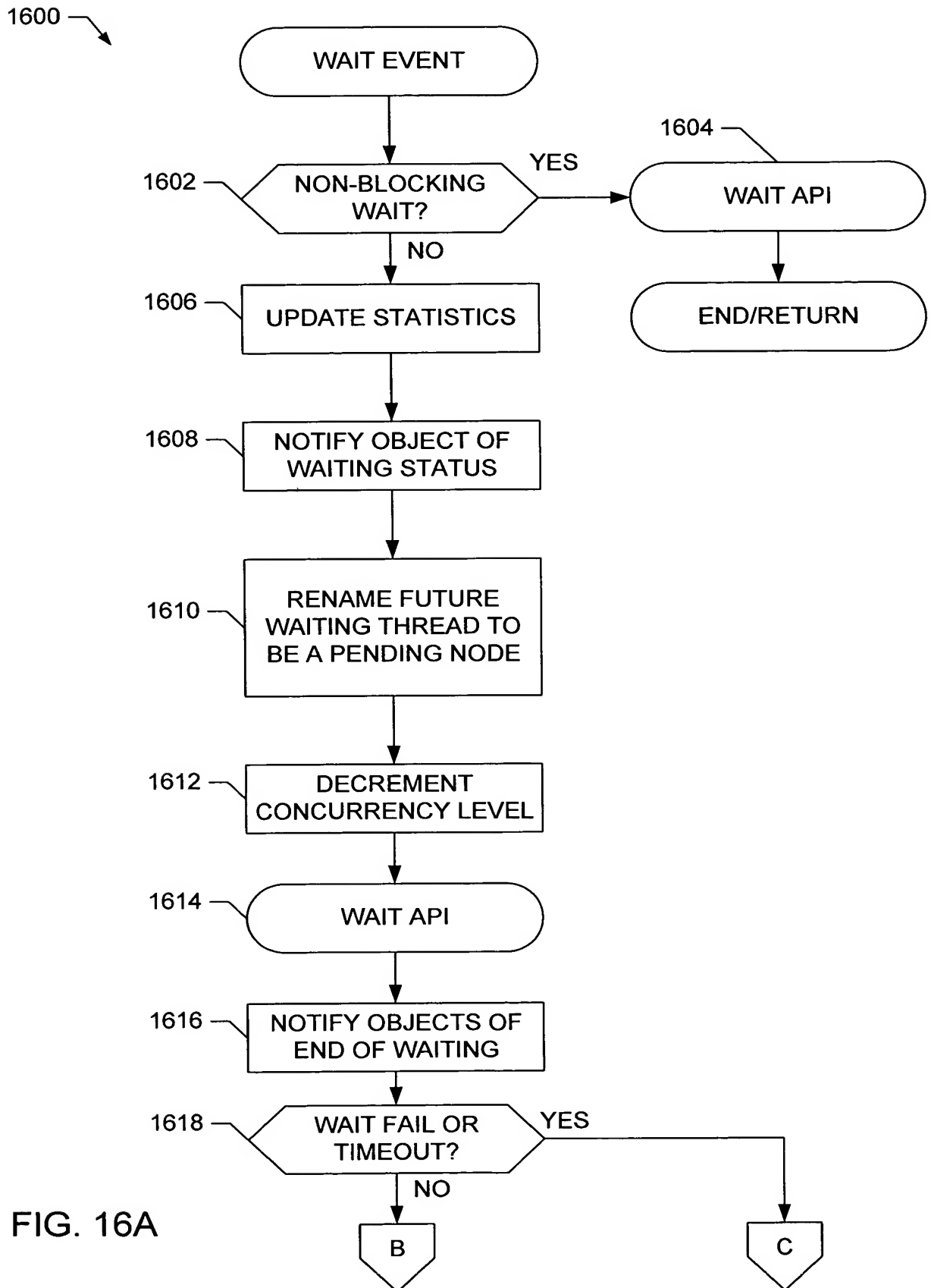
1500 

**WAIT**

```
* IF BLOCKING WAIT
- CHANGE WAITING THREAD STATE TO WAIT
- DECREMENT CONCURRENCY LEVEL
- FOR EACH OBJECT TO BE WAITED ON
  - REGISTER THAT THIS THREAD IS WAITING FOR THE OBJECT
    (ATOMIC INCREMENT WAIT COUNT OF OBJECT)
* DO THE API CALL
* IF NOT BLOCKING WAIT
  - DONE
* INCREMENT CONCURRENCY LEVEL
* FOR EACH OBJECT THIS THREAD WAITED ON
  - REGISTER THAT THIS THREAD NO LONGER WAIT FOR THE OBJECT
    (ATOMIC DECREMENT WAIT COUNT OF OBJECT)
* IF WAIT FAILED OR TIMED OUT
  - UPDATE LEAF OF CURRENT (WAITING) THREAD WITH TIME SPENT WAITING AS
BLOCKING TIME
ELSE /* WAIT DIDN'T FAIL */
  - FOR EACH OBJECT THIS THREAD WAITED ON
    /* CLAIM A LEAF FROM EACH OF OBJECT */
    - GET A PENDING NODE FROM THE OBJECT
    - IF THE RESOURCE COUNT OF NODE IS NOT INFINITE
      - DECREMENT RESOURCE COUNT
    - IF COUNT > 0
      - DUPLICATE THE PENDING NODE AND ADD TO A LIST OF POTENTIAL LEAVES
  - SELECT A POTENTIAL LEAF WITH A LATEST TIMESTAMP AND REMOVE THE REST
  - IF THE WAITING THREAD HAS A VALID RESUME LEAF (CREATED VIA A RESUME
ENTRY POINT) WHOSE TIMESTAMP IS LATER THAN THE CURRENT POTENTIAL LEAF
    - MAKE IT THE NEW POTENTIAL LEAF AND REMOVE THE OLD POTENTIAL LEAF
  ELSE
    - REMOVE THE RESUME NODE
  - IF THE WAITING THREAD'S PREVIOUS LEAF'S TIMESTAMP IS LATER THAN THE
CURRENT POTENTIAL LEAF
    - MAKE IT THE NEW POTENTIAL LEAF AND REMOVE THE OLD POTENTIAL LEAF
  ELSE
    - REMOVE THE OLD NODE
  - IF THERE IS A POTENTIAL LEAF
    - MAKE IT THE NEW LEAF FOR THE THREAD
  - IF THE THREAD'S NEW LEAF IS NOT THE THREAD'S OLD LEAF
    UPDATE STATS OF THE NEW LEAF
* SET THREAD TO ACTIVE
```

FIG. 15

13/17



14/17

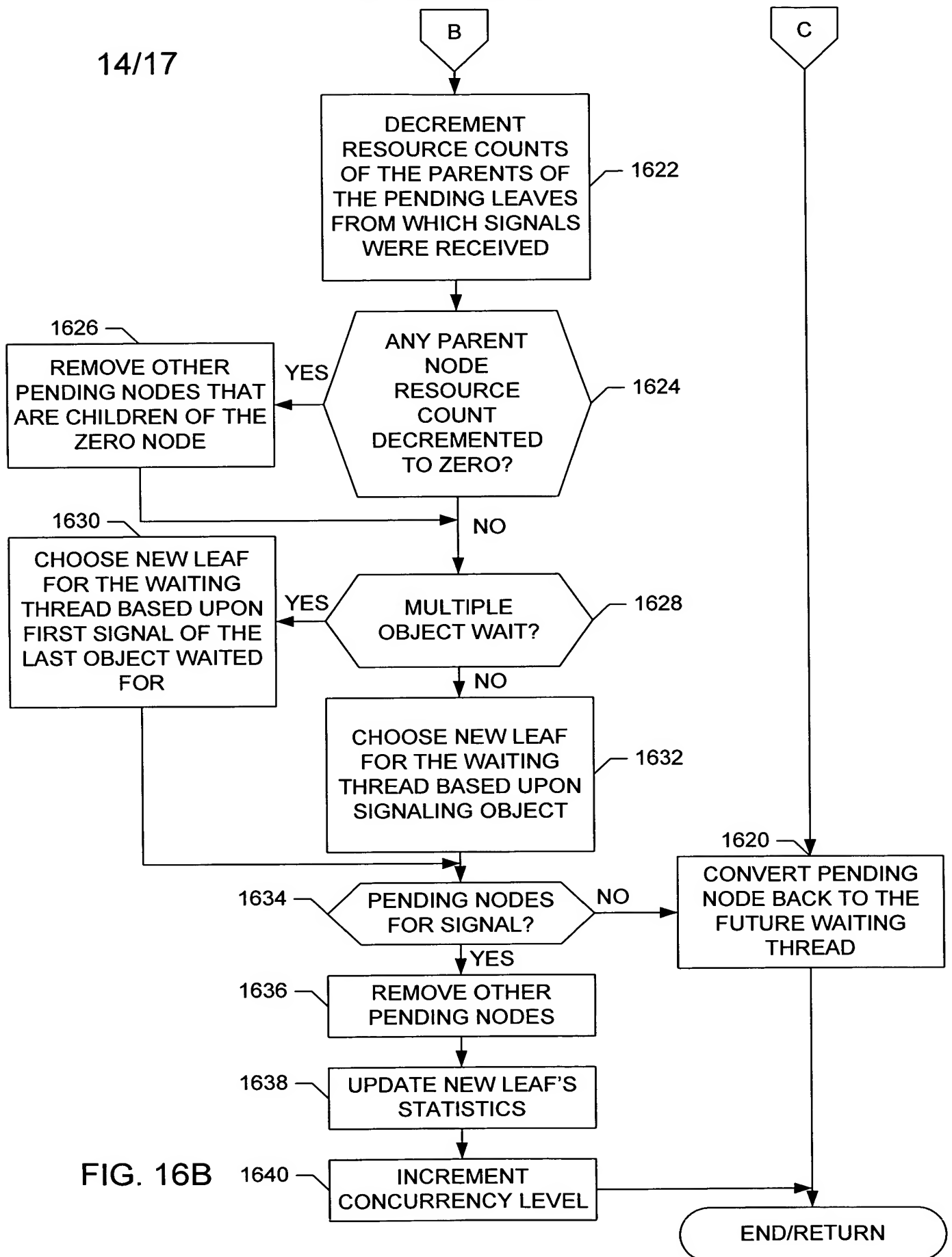
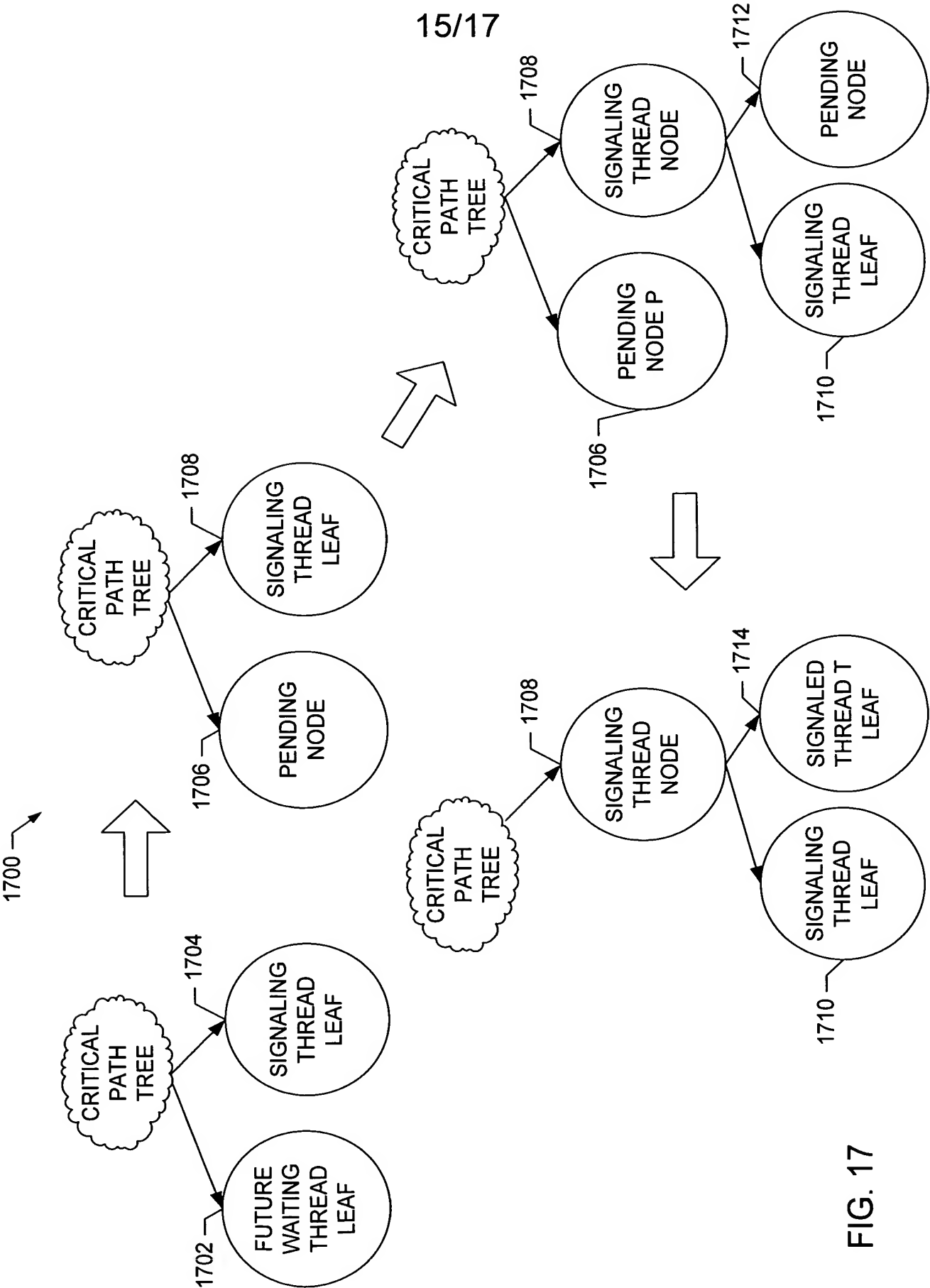


FIG. 16B



16/17

1800 ↘

### SUSPEND

- \* IF THE TARGET THREAD IS NOT ALREADY SUSPENDED
  - SET THE TIMESTAMP THAT THE THREAD IS SUSPENDED
- \* ACTUALLY DO THE API

FIG. 18

1900 ↘

### RESUME

- \* GET TIMESTAMP T BEFORE SIGNALING
- \* GET THE TIME THAT THE TARGET THREAD WAS SUSPENDED
- \* ACTUALLY DO THE API
- \* IF THE TARGET THREAD WAS NOT SUSPENDED
  - CLEAR SUSPENDED TIME OF TARGET THREAD
- ELSE IF THE TARGET THREAD WAS SUSPENDED BUT NOW RESTARTED
  - GET LEAF L OF THE RESUMING THREAD
  - CREATE A NEW LEAF FOR THE RESUMING THREAD W/ L AS PARENT NODE
  - CREATE A RESUME NODE FOR TARGET THREAD (WITH TIMESTAMP T) W/ L AS PARENT NODE
- REPLACE ANY OLD UNCLAIMED RESUME NODE OF TARGET THREAD WITH NEW NODE
  - IF AN OLD UNCLAIMED RESUME NODE EXISTS
    - REMOVE IT
  - IF THE TARGET THREAD WAS ACTIVE
    - USE THE TARGET THREAD'S NEW RESUME NODE AS ITS NEW LEAF & REMOVE OLD LEAF
  - UPDATE STATS OF THREAD'S NEW LEAF
  - SET TARGET THREAD STATE TO ACTIVE
  - INC CONCURRENCY LEVEL

FIG. 19

2000 ↘

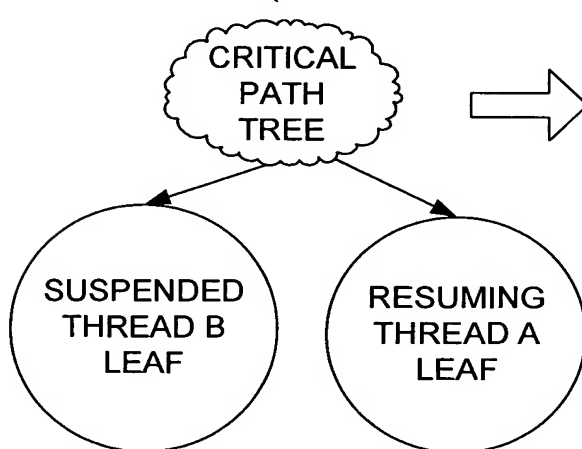
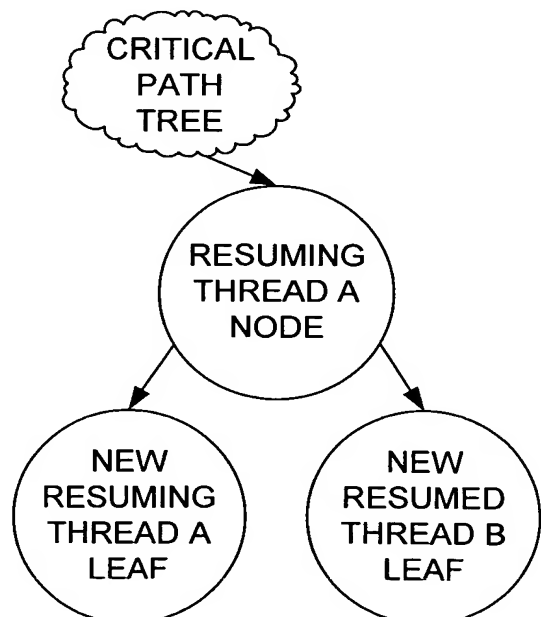


FIG. 20



17/17

2100 →

**BLOCK**

- \* SET CURRENT THREAD STATE TO BLOCK
- \* DECREMENT CONCURRENCY LEVEL
- \* DO API
- \* INCREMENT CONCURRENCY LEVEL
- \* IF THREAD NOW HAS A VALID RESUME LEAF (CREATED IN RESUME ENTRY POINT)
  - REMOVE CURRENT THREAD'S OLD LEAF
  - USE RESUME LEAF AS THE THREAD'S NEW LEAF
- \* UPDATE STATS OF THREAD'S LEAF
- \* SET THREAD STATE TO ACTIVE

FIG. 21

2200 →

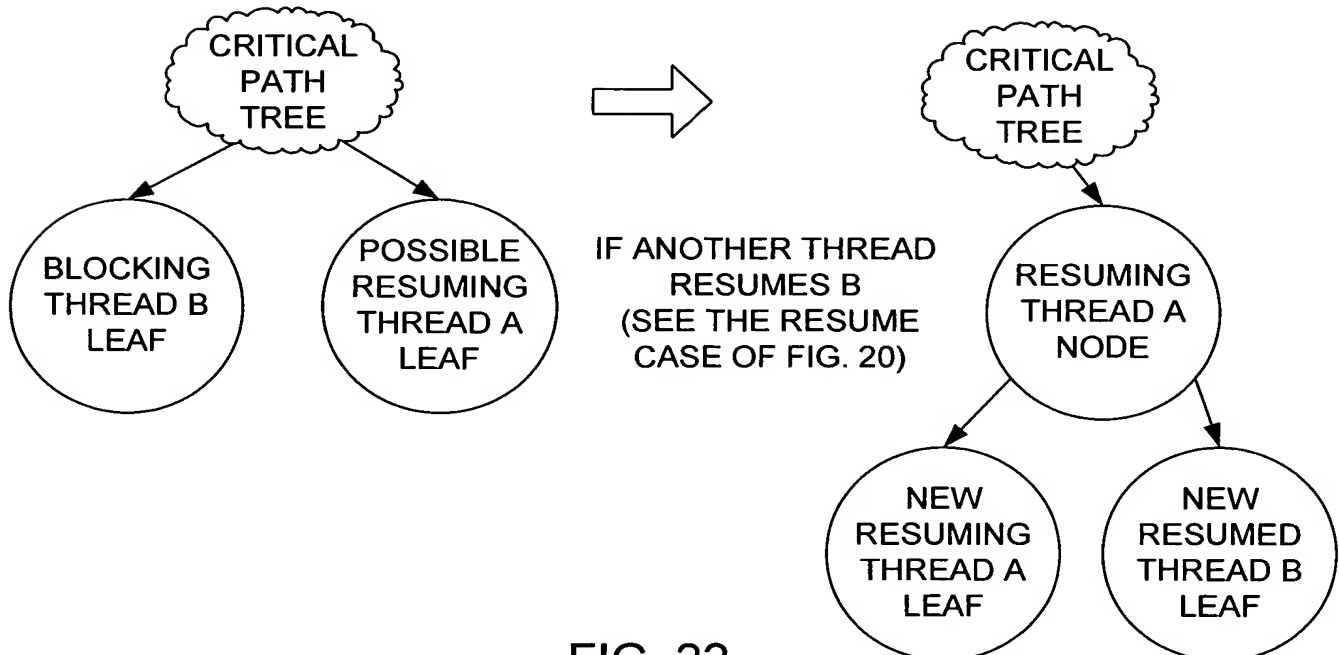


FIG. 22